K-mer Kernels for Genetic Distance

Chen Lu

May 5th 2016

Abstract

Phylogenetic tree reconstruction requires construction of a multiple sequence alignment (MSA) from sequences. However, alignment can be quite challenging for very large genomes. An interesting recent approach uses intensive hashing method to count the fraction of all unique k-mers appearing in either genome that are common to both to estimate the evolutionary distance, which avoids the need of assembly and alignment. In this project, I try to implement this k-mer idea using a more space efficient method based on the 2BWT. The program is written to process raw data, build 2BWT, count distinct and shared k-mers and generate the distance table. Using the simulated reads data from paper, I am able to construct the similar common k-mer table and phylogeny tree as the one in AAF paper. To further apply the method on large data set, 4 real genomes are tested and the resulting phylogeny tree looks the same as the one generated by the program phyloT.

Challenges

Phylogenetic analysis plays an essential role in revealing and understanding the history of evolutionary relationships. Many alignment methods and tools have been widely used for the construction of phylogenetic trees. Traditionally, a phylogenetic tree is constructed from multiple sequence alignment (MSA) of conservative proteins or genes. However, with the development of next generation technology, alignment can be very challenging for very large genomes, especially when the sequences are not collinear because one or both have undergone large-scale rearrangements [1]. Therefore, various alignment-free methods for phylogeny construction have been proposed, trying to circumvent the difficulty and the time consuming problem of constructing MSA for large genomes as well as the errors and inaccuracy introduced in the phylogenetic inference.

Existing Tools

Numerous multiple sequence alignment tools and alignment free tools for the phylogeny construction have been proposed and I will present a brief survey of some tools.
In alignment free methods, there are two main categories. The first category is based on k-mer/word frequency, such as Feature frequency profile (FFP), Composition vector (CV), Return time distribution (RTD), Frequency chaos game representation (FCGR) and Spaced-word frequencies, and the second category does not require resolving the sequence with fixed length word, such as average common subsequence (ACS), graph-based methods and the Kr estimator [1].

1. Clustalw [2]

Clustalw is a multiple sequence aligner that computes all pairwise distances between the input sequences. This is done either based on alignments or on a much faster alignment-free method.

2. AGP [1]
APG is a multimethods web server for alignment-free genome phylogeny construction. It implemented 12 most popular alignment-free methods for the construction of phylogeny trees using whole genomes and four methods for phylogenetic tree comparison.

3. Using the k-mer distance d_{kmer} [3,4,5]
One of popular alignment free method is based on kmer distance, which first proposed by Yang et al[3]. The k-tuple distance between two sequences is the sum of the differences in frequency, over all possible tuples of length k, between the sequences and can be estimated without MSAs. They showed in the paper that trees constructed from the k-tuple distance were more accurate than those from other distances most time, especially when the divergence between underlying sequences was high. In Fan et al.'s recent AAF paper [4], they also implement similar kmer distance idea.

Solving AAF with 2BWT

In the AAF paper, Fan et al. reconstructed the phylogenetic relationships among the genomes using pairwise distance matrix. They calculated the pairwise genetic distances between each sample using the number of evolutionary changes between the genomes that were represented by the number of different k-mers between genomes. The paper implemented a memory-intensive hashing approach to count shared k-mers [4].
In this project, using the same k-mer counting idea, I implement a more space-efficient method to build the distance matrix, which is based on the 2BWT, because BWT indexes offer significant improvements over hash-based methods in terms of both time and memory usage. 2BWT consists of the regular BWT along with the BWT of the reverse string, denoted rBWT. The regular BWT can be interpreted as the list of all left extensions of lexicographically sorted suffixes, whereas the rBWT can be interpreted as the list of all right extensions of colexicographically sorted prefixes. Algorithms based on the 2BWT maintain two intervals instead of one. First, the lexicographic range of the suffixes that are prefixed by the current substring, and second, the colexicographic range of all prefixes that are suffixed by the current substring. Compared with BWT, 2BWT allows both backward steps and left extensions as well as forward steps and right extensions, therefore, providing a way of symmetric index and operations. Therefore, 2BWT is widely used to solve more complicated problems in succinct space [6].

Implementation

A python script reverse.py was written to remove headers and unknown or masked nucleotides "N" in raw fasta files and concatenate the reads into one big string. It returned two files, one was the forward dna string in a *.fa file, the other was the reverse dna string in a *.rfa file. These two files could be later used to

generate corresponding BWT and rBWT using the JAVA program from homework2 coding project.
The whole phylogeny reconstruction process has two major steps: 1) k-mer counting and 2) distance calculation and phylogeny reconstruction. The python script shareKmer.py was written to count the shared kmers between two genomes as well as the distinct kmers for each genome. The distance between each pair of genomes was then calculated using the function from AAF paper [4].

$$D = -(1/k)^* \log(ns/nt)$$

The final phylogeny tree was drawn using online tool at http://www.trex.uqam.ca/index.php?action=trex&menuD=1&method=2.

The goal of the main algorithm is to count the intervals of all nodes that are at depth k shared in both sequences. The algorithm has two parallel stacks that always pop and push together, which ensures common k-mers. And we only pushed the intervals into stack if and only if both intervals generated from two sequences are valid. When the depth of the current node is equal k, we increment the count of share k-mers by 1 and continue to pop another stack. The algorithm works in time O(k∑).

Algorithm: Count all shared k-mers

```
Initialize empty stacks S1, S2
Count = 0
S1.push([1,n1],[1,n1],0)
S2.push([1,n2],[1,n2],0)
while S1 is not empty or S2 is not empty do
    (I1, d) = : top of S1 //always pop the stacks together
    (I2, d) = : top of S2
    If d == k: // k is the length of kmer
        Count++
        skip extendLeft, continue to pop another stack
    for all c ϵ ∑ do
        I1’ = ExtendLeft(I1, c)
        I2’ = ExtendLeft(I2, c)
        If I1’ or I2’ is not valid:
            skip push step
            continue to extend I1 and I2 of another char
        S1.push( I1’, d+1) // always push the stacks together
        S2.push( I2’, d+1) // push stacks only when both Is valid
```

Correctness
The correctness of my tool was tested on low complexity simulated data from the paper. As the paper stated, the primate genome assemblies were downloaded from the Ensembl database and the pair-end Illumina data was simulated using Dwgsim (Whole Genome Simulation, http://sourceforge.net/apps/mediawiki/dnaa/) assuming a read length of 70

bp, a sequencing error rate of 1 %, and coverages of 2X and 5X, with and without filtering.
Using the same simulated SRS read data, my AAF method obtained the same phylogeny for 10 primate species as the one in the paper (they had one more species which is rabbit), as shown in Figure 1.
Detailed k-mer counting table comparison was shown in Table 1. As we could see, the number of shared k-mers was similar under the two methods, though my method generated a little bit bigger number. For example, in AAF paper, there were 3882 shared k-mers between sp1 and sp2, whereas in the 2BWT method they shared 4094 common k-mers. One interesting fact was when I counted the distince k-mers for each genome, I was getting more than twice of the number as in AAF paper. But this should have trivial effect on the final distance table (Table 2) as long as all the nt are on the same level (i.e., all nt were around 165000 in AAF paper and were around 365000 in 2BWT AAF).

a

shared kmers from AAF Paper

	sp1	sp10	sp2	sp3	sp4	sp5	sp6	sp7	sp8	sp9
sp1	165116									
sp10	24379	164573								
sp2	3882	4093	165040							
sp3	24525	40234	4030	164817						
sp4	24323	37139	3851	37143	165735					
sp5	23813	39089	3750	55570	36149	164786				
sp6	24647	40273	4015	60322	37065	55948	164570			
sp7	58235	24242	3987	23978	23386	23222	24031	166312		
sp8	11519	11669	3630	11564	11113	11515	11358	11422	166712	
sp9	4828	4965	4352	5151	4940	5075	4966	4809	4820	164370

b

shared kmers from my program

	sp1	sp10	sp2	sp3	sp4	sp5	sp6	sp7	sp8	sp9
sp1	368448									
sp10	25237	367947								
sp2	4094	4287	368021							
sp3	25351	41503	4223	368238						
sp4	25156	38338	4067	38285	369291					
sp5	24691	40408	3958	57024	37290	368234				
sp6	25556	41434	4209	61729	38337	57281	367688			
sp7	59826	25076	4207	24819	24200	24084	24904	369401		
sp8	11984	12191	3813	12075	11641	12016	11897	11888	370211	
sp9	5086	5202	4565	5398	5192	5336	5226	5047	5059	368000

Table 1: shared kmers results from AAF Paper (a), my program (b)

distance table from my program

		Baboon	Orangutan	Bushbaby	Chimpanze	Gibbon	Gorilla	Human	Macaque	Marmoset	Ms_lemur
		sp1	sp10	sp2	sp3	sp4	sp5	sp6	sp7	sp8	sp9
Baboon	sp1	0	0.12760132	0.21421988	0.12742435	0.1278192	0.12867999	0.12696965	0.08656473	0.16312986	0.2038853
Orangutan	sp10	0.12760132	0	0.21201675	0.10391301	0.10769035	0.10518624	0.10395871	0.12790608	0.16224957	0.20280456
Bushbaby	sp2	0.21421988	0.21201675	0	0.21274258	0.21453497	0.21582863	0.2128576	0.21292334	0.21760589	0.20903164
Chimpanze	sp3	0.12742435	0.10391301	0.21274258	0	0.10779387	0.08882126	0.08497528	0.12843428	0.16274249	0.20105021
Gibbon	sp4	0.1278192	0.10769035	0.21453497	0.10779387	0	0.10904731	0.10765806	0.12977297	0.1646215	0.20290305
Gorilla	sp5	0.12867999	0.10518624	0.21582863	0.08882126	0.10904731	0	0.08853647	0.12986528	0.16297521	0.20160032
Human	sp6	0.12696965	0.10395871	0.2128576	0.08497528	0.10765806	0.08853647	0	0.1282003	0.1633785	0.20255184
Macaque	sp7	0.08656473	0.12790608	0.21292334	0.12843428	0.12977297	0.12986528	0.1282003	0	0.16363587	0.20425185
Marmoset	sp8	0.16312986	0.16224957	0.21760589	0.16274249	0.1646215	0.16297521	0.1633785	0.16363587	0	0.20413877
Ms_lemur	sp9	0.2038853	0.20280456	0.20903164	0.20105021	0.20290305	0.20160032	0.20255184	0.20425185	0.20413877	0

Table 2: Distance Table calculated from my k-mer results

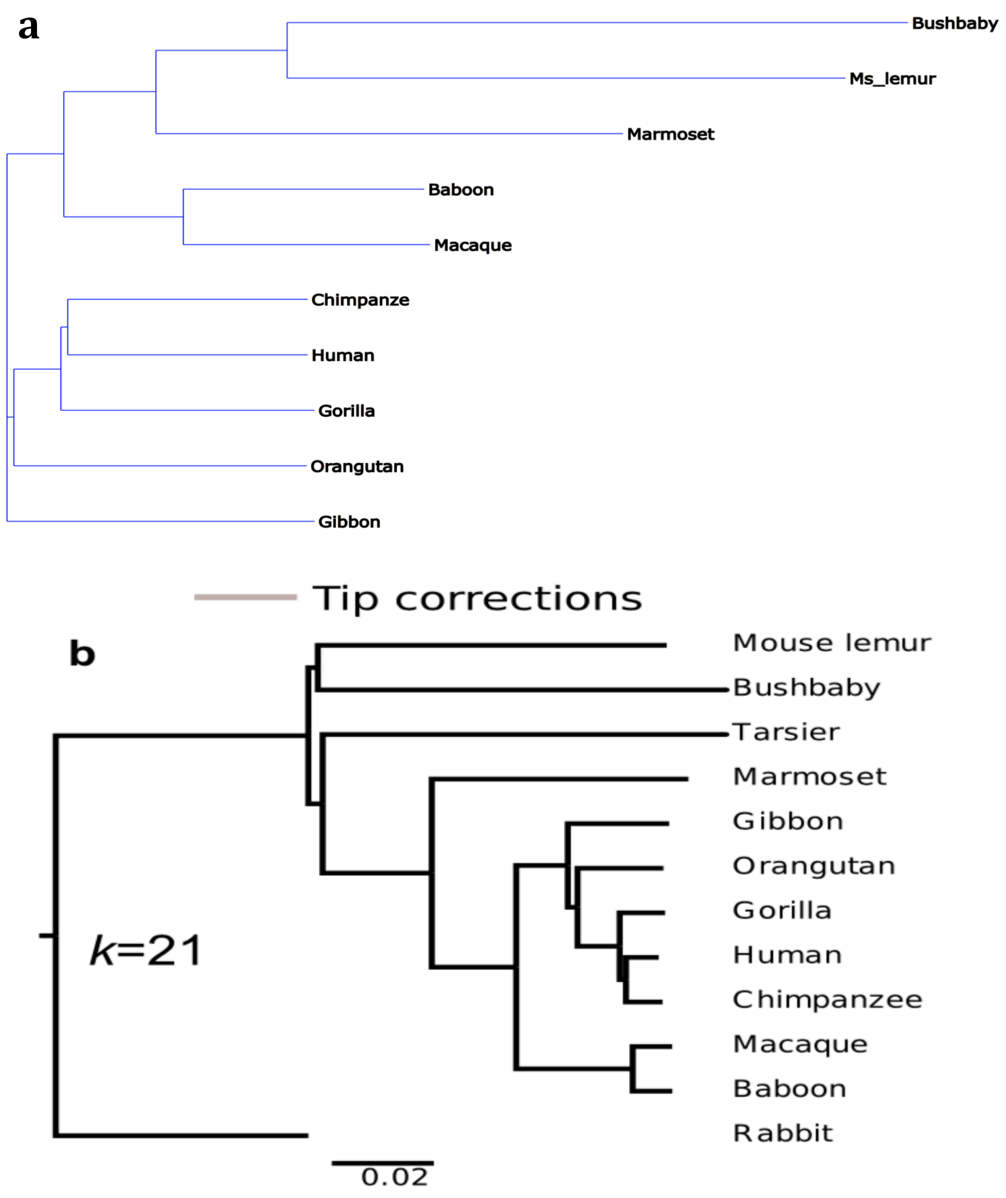


Figure 1: Phylogeny Tree built from distance table when k = 21 in my program (a), in AAF Paper (b).

Application on Large Data Set

To extend the 2BWT AAF application on large data set, I first reconstructed the distance table with amplified data set generated from the above small data set. A python script amplify.py was written to amplify each data for 250 times, followed by 2BWT construction and counting of shared k-mers. The resulting distance table and phylogeny tree looked exactly the same as the above.

Then, I further applied 2BWT AAF on 4 real genomes downloaded from Ensembl database. Their 2BWT files are listed:

C.savignyi.bwt	173.8 MB
C.savignyi.rbwt	173.8 MB
CaenorhabditisElegans.bwt	100.3 MB
CaenorhabditisElegans.rbwt	100.3 MB
CionaIntestinalis.bwt	112.2 MB
CionaIntestinalis.rbwt	112.2 MB
Fruitfly.bwt(DrosophilaMelanogaster)	142.6 MB
Fruitfly.rbwt(DrosophilaMelanogaster)	142.6 MB

I was planning to try larger data, however, the JAVA program for generating 2BWT was not able to handle files exceeding 250MB.
The 4 species phylogeny experiment was conducted on the Linux server provided by the WashU, which offers 16 cores and 128 GB of RAM. The program ran on separate servers for 6h.
The resulting K-mer table and distance table were shown in table 3. And the phylogeny tree drawn using the distance table looked the same as the one generated directly by phyloT online(Figure 2).

(a) K-mer Table

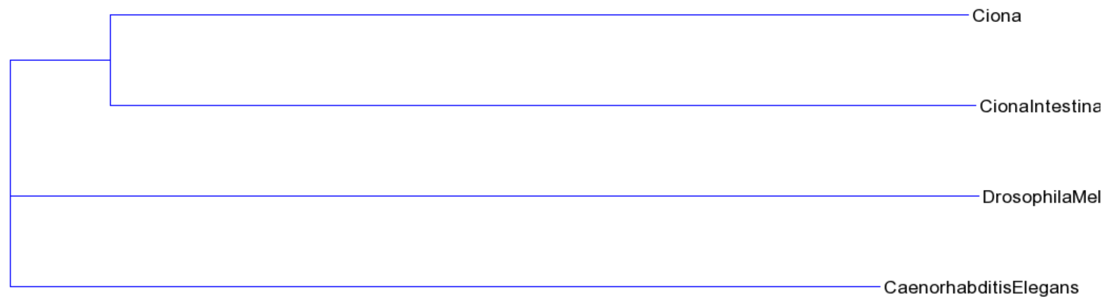
	CaenorhabditisElegans	DrosophilaMelanogaster	CionaIntestinalis	Ciona savignyi
CaenorhabditisElegans	93046064			
DrosophilaMelanogaster	115802	121944781		
CionaIntestinalis	111593	94003	102146777	
Ciona savignyi	126154	104930	193424	145931803

(b) Distance Table

	CaenorhabditisElegans	DrosophilaMelanogaster	CionaIntestinalis	Ciona savignyi
CaenorhabditisElegans	0	0.318522292	0.320285316	0.314445075
DrosophilaMelanogaster	0.318522292	0	0.332897112	0.336096674
CionaIntestinalis	0.320285316	0.332897112	0	0.298537208
Ciona savignyi	0.314445075	0.336096674	0.298537208	0

Table 3: K-mer Table (a) Distance Table(b) for the 4 species

a



b

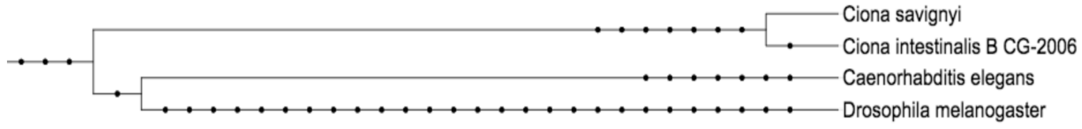


Figure 2: Phylogeny Tree built from my distance table when k = 21(a) generated by phyloT(b) for the 4 species

Bibliography

[1] Cheng J, Cao F, Liu Z. “AGP: a multimethods web server for alignment-free genome phylogeny. ” Mol. Biol. Evol. 2013; 30:1032–37.

[2] Jeanmougin F, Thompson J D, Gouy M. “Multiple sequence alignment with Clustal x” Trends Biochem. Sci. 1998; 23:403–405

[3] Yang K, Zhang L. “Performance comparison between k-tuple distance and four model-based distances in phylogenetic tree reconstruction.” Nucleic Acids Res 2008; 36(5): e33

[4] Fan, Ives, Surget-Groba, and Cannon, “An assembly and alignment-free method of phylogeny reconstruction from next-generation sequencing data” BMC Genomics 2015;16: 522

[5] Belazzougui and Cunial, “A framework for space efficient string kernels,” Proc. 2015 Conf. on Combinatorial Pattern Matching 2015; 26-39

[6] Jarno Niklas Alanko, “Space efficient clustering of metagenomic read sets”